Kotlin - Booleans

Many times we come across a situation where we need to take decision in **Yes** or **No**, or may be we can say **True** or **False**. To handle such situation Kotlin has a Boolean data type, which can take the values either **true** or **false**.

Kotlin also has a **nullable** counterpart **Boolean?** that can have the **null** value.

Create Boolean Variables

A boolean variable can be created using **Boolean** keyword and this variable can only take the values either **true** or **false**:

Example

```
fun main(args: Array<String>) {
   val isSummer: Boolean = true
   val isCold: Boolean = false

   println(isSummer)
   println(isCold)

}
```

When you run the above Kotlin program, it will generate the following output:

```
true
false
```

In fact, we can create Kotlin boolean variables without using **Boolean** keyword and Kotlin will understand the variable type based on the assigned values either **true** or **false**

Kotlin Boolean Operators

Kotlin provides following **built-in** operators for boolean variables. These operators are also called Logical Operators:

Operator	Name	Description	Example
&&	Logical and	Returns true if both operands are true	x && y
||	Logical or	Returns true if either of the operands is true	x || y
!	Logical not	Reverse the result, returns false if the operand is true	!x

Example

Following example shows different calculations using Kotlin Logical Operators:

```
fun main(args: Array<String>) {
   var x: Boolean = true
   var y:Boolean = false

   println("x && y = " +  (x && y))
   println("x || y = " +  (x || y))
   println("!y = " +  (!y))
}
```

When you run the above Kotlin program, it will generate the following output:

```
x && y = false
x || y = true
!y = true
```

Kotlin Boolean Expression

A Boolean expression returns either **true** or **false** value and majorly used in checking the condition with **if...else** expressions. A boolean expression makes use of relational operators, for example **>, <, >=** etc.

```
fun main(args: Array<String>) {
   val x: Int = 40
   val y: Int = 20

   println("x > y = " +  (x > y))
   println("x < y = " +  (x < y))
   println("x >= y = " +  (x >= y))
   println("x <= y = " +  (x <= y))
   println("x == y = " +  (x == y))
   println("x != y = " +  (x != y))
}
```

When you run the above Kotlin program, it will generate the following output:

```
x > y = true
x < y = false
x >= y = true
x <= y = false
```

```
x == y = false
x != y = true
```

Kotlin and()and or() Functions

Kotlin provides **and()** and **or()** functions to perform logical **AND** and logical **OR** operations between two boolean operands.

These functions are different from **&&** and **||** operators because these functions do not perform short-circuit evaluation but they always evaluate both the operands.

```
fun main(args: Array<String>) {
    val x: Boolean = true
    val y: Boolean = false
    val z: Boolean = true

    println("x.and(y) = " +  x.and(y))
    println("x.or(y) = " +  x.or(y))
    println("x.and(z) = " +  x.and(z))
}
```

When you run the above Kotlin program, it will generate the following output:

```
x.and(y) = false
x.or(y) = true
x.and(z) = true
```

Kotlin also provides **not()** and **xor()** functions to perform Logical **NOT** and **XOR** operations respectively.

Boolean to String

You can use **toString()** function to convert a Boolean object into its equivalent string representation.

You will need this conversion when assigning a **true** or **false** value in a String variable.

```
fun main(args: Array<String>) {
    val x: Boolean = true
    var z: String

    z = x.toString()

    println("x.toString() = " +  x.toString())
    println("z = " +  z)
}
```

When you run the above Kotlin program, it will generate the following output:

```
x.toString() = true
z = true
```

Quiz Time (Interview & Exams Preparation)

Q 1 - Which of the following is true about Kotlin Boolean Data type?

A - Boolean data type can have two values true and false

B - Boolean data type can have two values 0 and 1

C - We can assign Boolean value to integer variable

D - All of the above

Q 2 - What will be the output of the following program:

```
fun main(args: Array<String>) {
    val x: Boolean = true
    var y: String

    y = x
}
```

A - This will compile successfully without error and warning

B - This will raise just a warning

C - Compilation will stop with error

D - None of the above